####笔记
C++_primer_笔记

==========updating==========

####c调用c++
如何用C语言封装 C++的类，在 C里面使用 - 小猪爱拱地 - 博客频道 - CSDN.NET

####
C++笔记（1）explicit构造函数 - cutepig - 博客园
explicit　只对构造函数起作用，用来抑制隐式转换。

####
关键字 mutable (c++) - typhoonzb的专栏 - 博客频道 - CSDN.NET
1.一个对象的状态由其非静态数据成员的值构成，因此，修改一个数据成员将会改变整个对象的状态。将一个成员函数声明为 const 能够保证它不会改变对象的状态。
2.对C++ 了解的人都应该知道虚函数（Virtual Function）是通过一张虚函数表（Virtual Table）来实现的。简称为V-Table。 在这个表中，主是要一个类的虚函数的地址表，这张表解决了继承、覆盖的问题，保证其容真实反应实际的函数。
3.在C++的标准规格说明书中说到，编译器必需要保证虚函数表的指针存在于对象实例中最前面的位置（这是为了保证正确取到虚函数的偏移量）。
4.通过vTable获取虚函数地址，调用虚函数：可以通过强行把&b转成int *，取得虚函数表的地址，然后，再次取址就可以得到第一个虚函数的地址了，也就是Base::f(),

```
class Base {
public:
virtual void f() { cout << "Base::f" << endl; }
virtual void g() { cout << "Base::g" << endl; }
virtual void h() { cout << "Base::h" << endl; }
};
按照上面的说法，我们可以通过Base的实例来得到虚函数表。 下面是实际例
typedef void(*Fun)(void);
Base b;
Fun pFun = NULL;
cout << "虚函数表地址：" << (int*)(&b) << endl;
cout << "虚函数表 — 第一个函数地址：" << (int*)*(int*)(&b) << endl;
// Invoke the first virtual function
pFun = (Fun)*(int*)*(int*)(&b);
pFun();
```

5.函数表示意：

```
(Fun)*((int*)*(int*)(&b)+0); // Base::f()
(Fun)*((int*)*(int*)(&b)+1); // Base::g()
(Fun)*((int*)*(int*)(&b)+2); // Base::h()
```

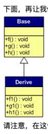
这个时候你应该懂了吧。什么？还是有点晕。也是，这样的代码看着太乱了。没问题，让我画个图解释一下。如下所示：

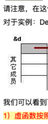
注意：在上面这个图中，我在虚函数表的最后多加了一个结点，这是虚函数表的结束结点，就像字符串的结束符"\0"一样，其标志了虚函数表的结束。

6.一般继承(无虚函数覆盖)：

一般继承（无虚函数覆盖）

下面，再让我们来看看继承时的虚函数表是什么样的。假设有如下所示的一个继承关系：





请注意，在这个继承关系中，子类没有重载任何父类的函数。那么，在派生类的实例中，其虚函数表如下所示：

对于实例：Derive d; 的虚函数表如下：

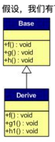
我们可以看到下面几点：

1）虚函数按照其声明顺序放于表中。
2）父类的虚函数在子类的虚函数前面。

7.一般继承（有虚函数覆盖）

一般继承（有虚函数覆盖）

覆盖父类的虚函数是很显然的事情，不然，虚函数就变得毫无意义。下面，我们来看一下，
假设，我们有下面这样的一个继承关系。



为了让大家看到被继承过后的效果，在这个类的设计中，我只覆盖了父类的一个函数：f()。

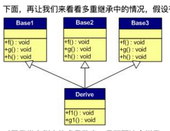
我们从表中可以看到下面几点，

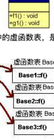
1）覆盖的f()函数被放到了虚表中原来父类虚函数的位置。
2）没有被覆盖的函数依旧。

8.多重继承（无虚函数覆盖）

多重继承（无虚函数覆盖）

下面，再让我们来看看多重继承中的情况，假设有下面这样一个类的继承关系。注意：子类并没有覆盖父类的函数。





对于子类实例中的虚函数表，是下面这个样子：

我们可以看到：

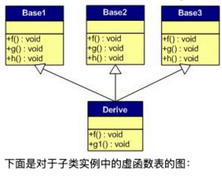
1） 每个父类都有自己的虚表。
2） 子类的成员函数被放到了第一个父类的表中。（所谓的第一个父类是按照声明顺序来判断的）

这样做就是为了解决不同的父类类型的指针指向同一个子类实例，而能够调用到实际的函数。

9.多重继承（有虚函数覆盖）

多重继承（有虚函数覆盖）

下面我们再来看看，如果发生虚函数覆盖的情况。
下图中，我们在子类中覆盖了父类的f()函数。



下面是对于子类实例中的虚函数表的图：

我们可以看见，三个父类虚函数表中的f()的位置被替换成了子类的函数指针。

####
编译错误 --- does not name a type和field `XX' has incomplete type - guogaofeng1219的专栏 - 博客频道 - CSDN.NET
B类中A类类型的成员变量，class B定义声明之前先声明一下class A；并且将class B定义中的A域改用指针就行了。

####
c++ class does not name a type - typename 记录点滴 - 博客频道 - CSDN.NET
使用前置声明时，cpp文件中include 头文件次序必须先 包含前置声明的类定义头文件，再包含本类头文件。
否则会出现如下编译错误.
(expected　constructor, destructor, or type conversion before ‘typedef’)